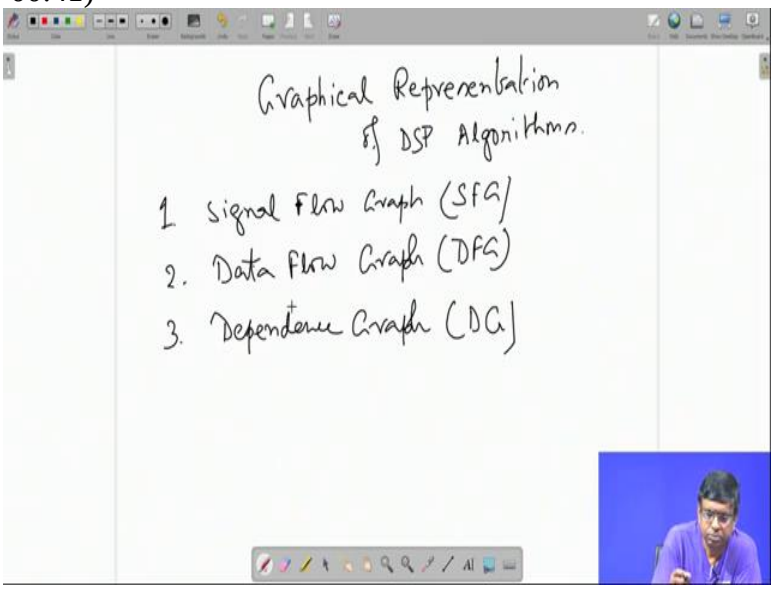
Course on VLSI Signal Processing
Professor Mrityuunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture 01
Graphical Representation of Signals

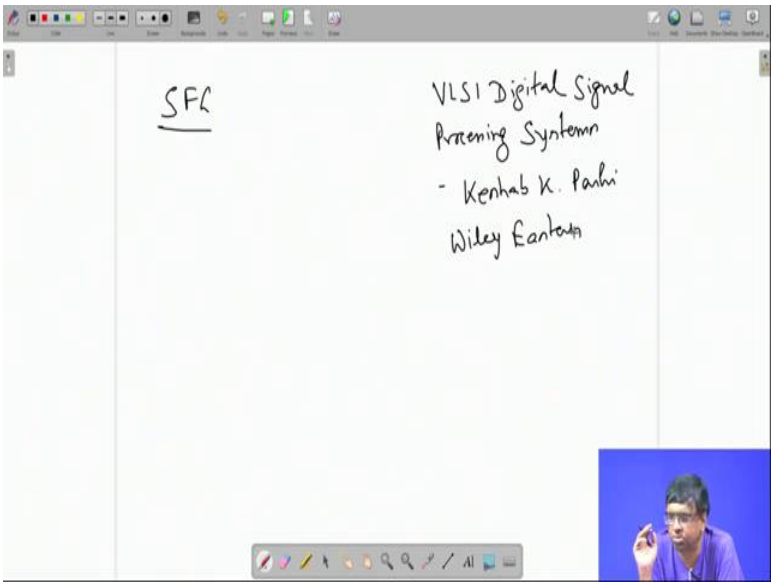
(Refer Slide Time: 00:41)



Ok, so we begin this course on VLSI Signal Processing. I had already given preview of what the course content should be. So, I am not getting into that, straight I will start with the first topic, which is graphical representation of DSP algorithms. We have got 3 kind of graphs, one is called signal flow graph that is SFG. Next is data flow graph DFG and third is dependence graph.

Of these three, the first one is not very widely used but then you have to start with these to understand what data flow graph DFG and the dependence graph ok. So, we did our discussion with signal flow graph.

(Refer Slide Time: 02:05)

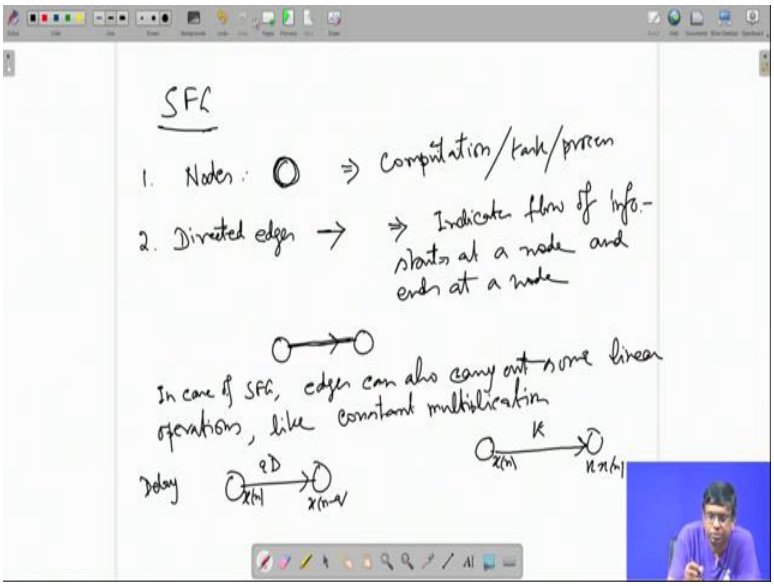


By the way, I forgot to mention you should for most of my lectures, it is very important that you follow my lectures. Because I teach from my own understanding, you will not find normally books, but still indeed your reference, for that you should look at this book by Keshab K. Parhi it is available in Indian addition Wiley Easton.

So, very a cheap book this Keshab Parhi has been an ex-student of IIT Kharagpur and has become very famous, he is a very well-known professor at university of Minnesota, Minneapolis. He has come out with this book and this is a unique book of its kind because we do not have any other book like this available. Which this is see so many types of you know VLSI architecture optimization techniques for DSP and communications, alright.

So many results are available but they are all scattered in journals in various forms, but it goes through his credit to compile all of them together and I mean friends and together and come up with a very nice book of course this is not an easy book, this is a very difficult book to read. That is why I come in, if you follow my lectures and also content, the book that things will be easy. Also many things I teach which are not in the book at all. So, you have to be very careful to, while attending my lectures, ok. It is very important that you follow my lectures and then you also consult this book all right.

(Refer Slide Time: 04:10)



Single flow graph, this consists of two items mostly, one is called node they are denoted by bubble. This indicates, nodes indicate computation of some task process whichever you want to call it computation that is it does some job. Very several jobs many jobs can be put under one node or maybe a simple operation can be under one node.

It is all up to you, to decide the about what will be a node. What was a operations will be within a node but node is indicated by a bubble. Another is called directed edges, indicated by this. It means it indicates flow of, indicates flow of information. I mean it starts at a node, starts at a node, and ends at a node like this.

The two nodes, there can be an edge like this, which means there is a physical connection from this node to this node through a line. But this line the arrow indicates the direction that is why it is called directed edge. And these directed edge gets information can only come from this node and go up to these node not in the opposite direction.

So, these are what edge and direction indicates flow of direction which means a physical connection, there will be a physical connection (())(06:14) which will take information data from this node and take it to the other node, alright. Now, in the case of a SFG, in case of a SFG node edges can also carry out some linear operations like constant multiplication. That is we have an

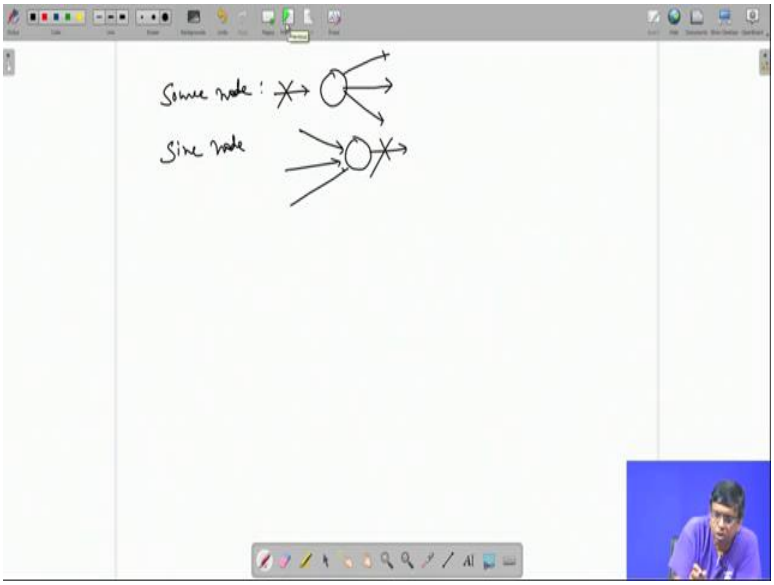
edge, if I by the side of it write K, it will be whatever information coming out of this node that will be multiplied by K and they will reach here.

So, if that this node produces some sequence xn what will come here is K xn. So, this multiplayer which is a linear operation this we can write by the side of the edge, so in the case of SFG edge not only transmits information, but also within the during that time it can carry out a linear operation like Mccosker multiplication. Remember how do you delete the other graphs DFG and DG.

There also we have nodes and edges like this absolutely same, but they are the edges may not be, may not be able to carry out linear operations like this. In the case of SFG constant multiplication is possible so you write K by the side of an edge it will be K times incoming information that will be given to the out of this node the destination node. Another operation is delay.

That is suppose if I have got things like this and I write q times D, D is delay, 1 D means 1 clock cycle delay, 2 D means 2 clock cycle delay, q D means q clock cycles delay. This will be if the node generates subsequent xn this will be delayed by q cycles and what will come here will be xn minus q. So, in the case of SFG you can have edges, directed edges not only carrying transfer of information, but also you can have constant multiplication or delay you know along with the edge, that will be also part of the edge. That is true only of SFG not true of the other two graphs.

(Refer Slide Time: 09:38)

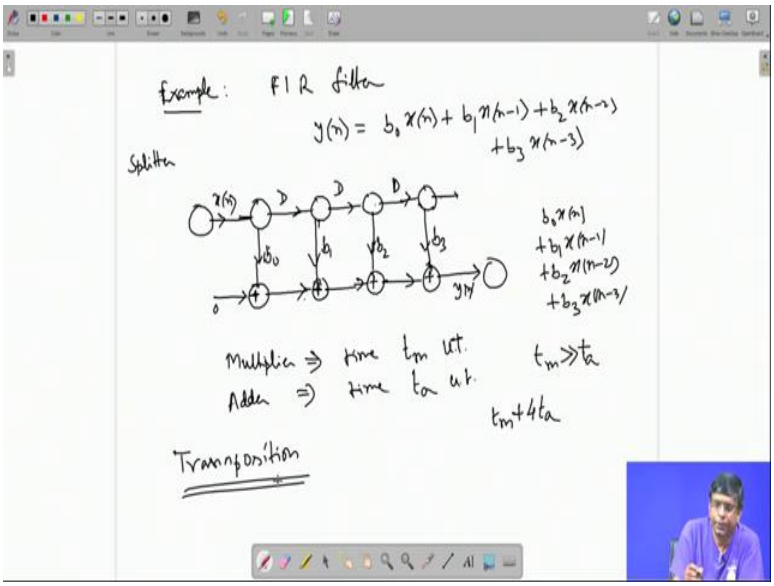


And there is a source node it is like this, a node from which edges only come out no edge comes to it from any other node. It is like a signal generator it only generates data information signal and then it goes to other nodes like here, even a multi edges we got but nothing comes in. Nothing comes in. So, source node similarly you can have a sink node, here there can be many incoming edges falling on it, but nothing goes out, nothing goes out that is called sink node. So it is a terminating nodes, both the terminal nodes, 1 is at the beginning is at the end, alright.

And now, when I give you problems again DSP algorithm and I asked you to draw a signal for graph or DBDG or dependency graph or data flow graph, one thing I must make clear at this stage there is nothing unique.

There is nothing unique about the graph, the graph depends on you. I mean, what all operations you are putting in a node and how are the nodes getting connected, it all depends on your choice. Of course, we would stick to certain choices because there is optimal, but again I tell you, there is nothing unique about it. It is depends on who is drawing the graph how he perceives, alright.

(Refer Slide Time: 11:11)



So, with this I take an example FIR filter. Suppose you have got yn is a filtered output, b0 times Xn, b0 first coefficient, then b1 times xn minus 1, then b2 times xn minus 2, and b3 times xn minus 3.

There is a 4 coefficient, third order because (())(11:59) we will have up to get to the power 3, from this last term. There is a third order filter or equivalently four terms, four multipliers. I want

to draw a DG. So, obviously I am sorry, SFG singular program. So, I will have a source node that will be generating xn. Then did I tell you it all depends on how you draw it depends on the person I m drawing.

So I will draw in a particular way. I will have a node here this node is called splitter node. What it does is this, it takes the incoming data and split I mean gives it out into the two directions. The same xn comes out in this direction also these direction also. So it splits, they by the side of it is an edge. I can have a multiplier or a delay or both side at b0. There are another node normally, these 2 can be defined nodes but we indicate them by bubbles or cycles.

But then one can ask the question how do I do know what do kind of node it is and what kind of node this is, just for just to see that you are not confused I put a plus here to indicate it is an adder node. Adder means, it takes 2 input 1 output there is an added so this is 1 input. Another one I give 0 so it is an edge. With a splitter and here I put 1 cycle delay 1 into D 1 cycle delay, then I get splitter node, it will split the same thing in two directions and this edge, sorry. I get a plus.

So, this edge the single gets multiplied by b1, this signal gets multiplied by b0. Again an adder this goes to a splitter I will tell you what we are doing actually. But by the side of this edge I give 1 cycle delay, so 1D which is D again this splitter it splits this single in 2 this is splitter node, so it splits it into two directions, one is this direction and there is again an adder node. And I give a multiplier with this b2 and this goes to another node.

At with one cycle delay on this goes this way. There is another edge, this edges are free now no multiplier with them, no delay with them. But node to node, b3 with this and there is a sink node. So, what is happening you see xn it is going here it is getting split into two direction this is coming as xn multiplied by b0. So, you have b0 the xn plus 0. So, I have b0xn here b0xn, this is getting delayed by 1 cycle.

So, I have what b0xn here. Now, xn coming out of this node in this direction also because this is a splitter node that is getting delayed by 1 cycle so xn minus 1. These is again a splitter node this will come in this direction multiplied by b1 added with this. So, I have got (())(16:03) this summation moves xn minus 1 is further delayed in these xn, xn minus 2 get split into two directions, this way it is gets multiplied by b2 gets added with this.

So, b2xn minus 2 and this gets further delayed by 1 cycles it becomes Xn minus 3 it comes to this direction b3xn minus 3 gets added with this, this was outputted clearly this is the yn. So, these are simply SFG. The dink node source node I do not have to do anything with these if I want to have more and more stages, this will get continued. Now one thing you have to calculate, suppose, adders we have got two types of jobs here.

There are four multipliers in parallel along the edge I have b0, b1, b2, b3 they are working in parallel, (())(16:57) multipliers the incoming data here b1 multiples the incoming data coming from this node b2 multiplies the incoming data b3. So, suppose multipliers it takes time tm maybe tm microsecond or pico second or nanosecond, whatever the unit you deploy. I write u dot t dot unit of time, tm unit of time.

And then I have got adders, splitters do not take any time whatever is coming, you are just passing it in two directions. So, that does not require any time but multiplications yes, they take a lot of time in VLSI design multipliers are big headaches because how do you multiply into 32 bit numbers, you have got 32 by 32 array shift that, shift that, shift and go on doing a 32 times, 31 times, so that requires a lot of time, a lot of hardware that is why multipliers are the biggest concern in VLSI for VLSI designers we try to either minimize multiplication complexities or get rid of multipliers altogether if possible.

Same is not for adders, adders are much similar. So, adder the time I call it ta unit of time. Again ta microsecond, millisecond I mean nanosecond, pico second whatever the time involved. Obviously tm must be greater than ta, with multipliers take much more time than adders. The question is the question that will ask is this if I give an input xn, how much time will be required to generate yn before I can give next data?

Because I have to give this entire circuit some time to compute the job there is to compute yn then only I should give the next one, right. How much time? Now you see as a I told you splitters they do not take any time. So, here I have got 4 multipliers in parallel b0, b1, b2, b3. So, they are working in parallel means, they occupy the same time, during the same time four are working parallely.

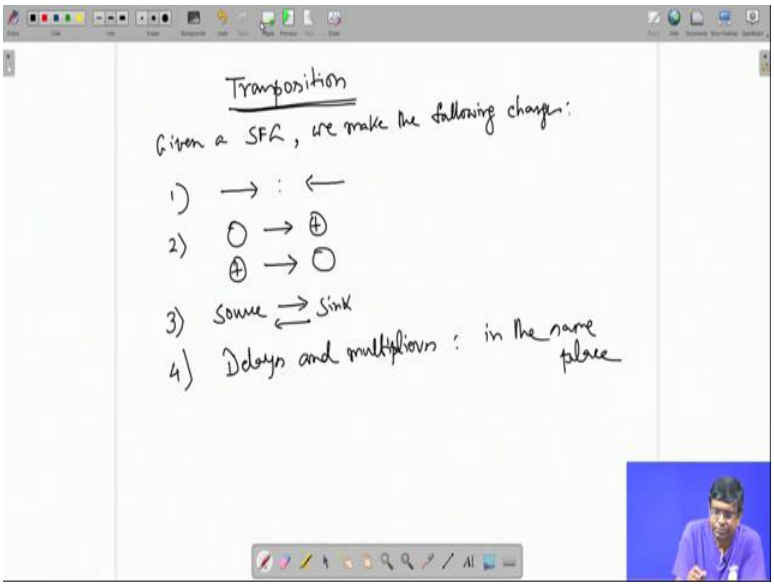
So, I will have tm but how about the adders, adders have a problem these adders takes ta amount of time, but it adds this guy and this guy, but this guy is not available unless these adder completes the job of addition, then only this available. So before this ta, another ta amount of time is to be employed here to compute this addition and again, this is, this can be obtained only if this addition is over.

So, another ta amount of time needs dedicated to do this job before I go to this, and again this requires this which is obtained from this addition. So, I need another ta amount of time. So, total I need 4ta. This much amount time, if I got instead of 4 I have got 400 terms I would have tm plus 400 ta. Multipliers are working parallelly, so tm (())(20:02) tm, but all the adders you see, they are sequential, one adders output goes to the input of the other adders.

So, first you have to complete the job then I go here, then again this goes to input of other adder there is they have to complete this then only next adder can start operation, so on and so forth. So, the adders are sequentially connected here one after another that is why the all the times get added the while. So, some transformation technique by who is this given architecture will be changed to transform to another architecture, which is equivalent that is where if I give the same input xn, I would get the same output yn but there will see the time required to compute yn for an xn will be lesser than this, which means that circuit will be faster it can handle data at faster rate because its own computation time overall computation time will be less than this.

There is a called transpose form, that is I mean we obtained by technical transposition. So, there are many high level transformation techniques that give enough graphical description you can apply one of the transformation techniques, get something which is equivalent to this in terms of the input output relation for the depth, equivalent circuit we have a better feature than this maybe in terms of speed, maybe in terms of total computation and hardware or cheap period so and so.

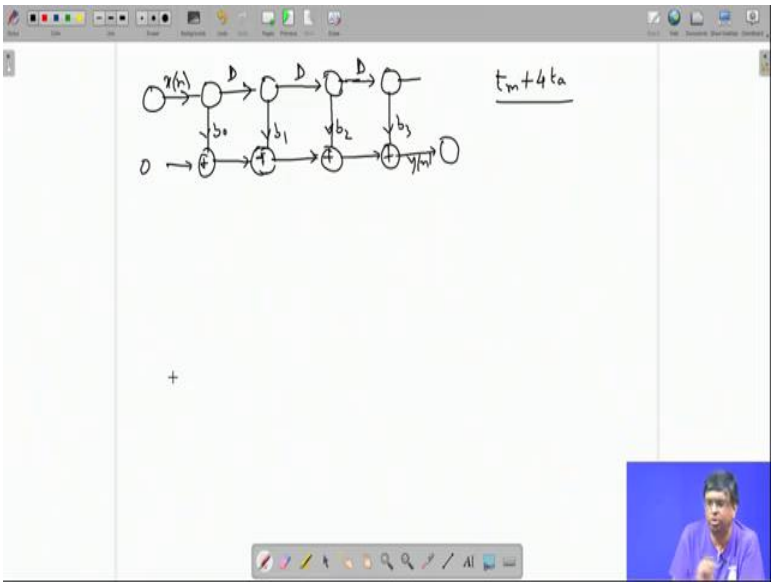
(Refer Slide Time: 22:00)



In the case of single processing graph, there is one technique called transposition. In the next class, in the next topic I will discussed transposition of SFGs. We are given the SFG, we make the following changes I wonder whether we can do it in the next class or we will continue here. You reverse the arrows, if there is an arrow pointing in one edge, keep the edge in the same place but reverse the direction in the SFG.

Number 2, splitter node change it to adder and adder changing to splitter. Source and sink interchange them, source and sink. And lastly delays and multipliers in the same place, do not disturb. If we apply these techniques to a SFG then you get a transpose form. Then you get it transpose from this is SFG. Alright, will apply these techniques to that SFG of FIR filter we have obtained a while ago.

(Refer Slide Time: 24:23)



So, I once again draw that SFG, so we had that this time source. Once I draw these I will stop now. And then I will take it up in the next class. We had already drawn it but for computation purpose I had to still draw it again. So, I had only drawn it last time. And it goes to source sink node. Alright b1, b2, b3 and this and time taken was as I found that time tm plus 4ta, this much time was taken.

Now we will be applying transposition to this circuit and we will get an equivalent form. We will first see that the one which you get by your transposition is indeed equivalent to these. That is if I give you xn and get yn here in the transform circuit also, if I give xn at the input I will get back yn and then I will try to calculate the time taken by the circuit to produce the output yn for a given xn and that time we will see it will be less like this which means as far as speed is concerned that will be faster more preferable, but that will have his other demerits also that also we will discuss. So, that is all for this class will move to the next class and then we will discuss this. Thank you.